Utilisation des chaînes d'automates pour le diagnostic décentralisé

Using automata chains for the decentralized diagnosis

Alban Grastien
Université Rennes 1 – Irisa
Avenue du Général Leclerc
35 042 Rennes Cedex – France
E-mail : alban.grastien@irisa.fr

Résumé

Dans le domaine du diagnostic de systèmes à événements discrets, la reconnaissance de la concurrence dans les comportements de sous-systèmes permet de considérer ceux-ci de manière indépendante et d'empêcher l'*explosion du nombre d'états* lors du calcul du diagnostic. Cependant, lorsque l'on considère des périodes de longueur importante, il est rare d'obtenir un comportement qui soit concurrentiel pendant toute la période. Nous proposons d'utiliser le découpage d'automate appliqué au diagnostic pour découvrir des périodes de comportements indépendants et éviter l'explosion combinatoire.

Mots Clef

Systèmes à événements discrets, diagnostic, diagnostic décentralisé, chaîne d'automates

Abstract

The decentralized diagnosis enables to cope with the so-called *state explosion problem* by using the properties of concurrency in the behaviors of sub-systems. However, when considering long periods, the behaviors of the sub-systems are rarely independant from the beginning to the end. We propose to use the slicing of automata applied to diagnosis to find independant behavior periods and to cope with the state explosion problem.

Keywords

Discrete event systems, diagnosis, decentralized diagnosis, automata chain

1 Introduction

Dans le domaine des systèmes à événements discrets (SÉD[1]), le comportement du système est souvent représenté par un automate *Mod*. Cet automate décrit les évolutions possibles du système vis-à-vis de stimuli extérieurs (événements exogènes) ainsi que les observations émises par le système lors de l'occurrence de ces événements. Le diagnostic du système consiste alors à calculer les *trajectoires* sur le modèle expliquant les observations émises par le système, c'est-à-dire en synchronisant le modèle avec les observations émises [2, 3]. Lorsque l'on considère que les observations émises ne sont pas trivialement reçues (délais entre l'émission et la réception des observations, non synchronisation des capteurs, pertes possibles, etc.), on ne connait pas précisément la séquence des observations émises. On peut cependant les représenter sous la forme d'un automate *Obs*. Le diagnostic se définit alors comme la synchronisation des automates représentant le modèle et les observations : $\Delta = Mod \otimes Obs$.

Dans les systèmes réels, le modèle est de taille souvent trop importante pour être calculé. De même, le diagnostic est généralement très grand notamment à cause des comportements concurrentiels. Pour cette raison, l'approche décentralisée [4] a été développée. Dans cette approche, on considère que le système est constitué d'un ensemble de composants (physiques ou abstraits) interreliés qui communiquent par l'envoi de messages synchrones. Chaque composant dispose d'un modèle. Ce modèle décrit l'évolution du composant lors de l'occurrence d'un événement exogène ou de la réception d'un message provenant d'un autre composant. Ainsi, le modèle *global* du système (généralement de taille exponentielle par rapport au nombre de composants) ne peut pas être construit mais peut être implicitement obtenu par synchronisation des modèles locaux. De plus, l'approche présentée permet de construire un diagnostic localement à chacun des composants, puis de fusionner les diagnostics pour vérifier les interactions entre les composants. Lorsque la fusion des diagnostics de sous-systèmes n'apporte plus d'information parce que les sous-systèmes considérés ont un comportement indépendant (ils ne communiquent pas entre eux durant la période), on arrête la fusion et on évite ainsi la complexité trop importante du diagnostic.

L'un des problèmes de la technique de diagnostic décentralisé est que les différentes parties d'un système n'évoluent indépendamment que durant de courtes pé-

riodes, et que lorsqu'on cherche à diagnostiquer un système dans le monde réel, la période à diagnostiquer est si longue qu'il n'est pas possible de trouver des sous-systèmes indépendants. C'est pourquoi, nous proposons d'utiliser les chaînes d'automates pour découper la période de diagnostic en plus petites périodes durant lesquelles il est possible d'exhiber des comportements indépendants.
La section suivante présente les définitions classiques d'automates utilisées dans le reste de cet article. Dans la section 3, le principe du diagnostic décentralisé et ses limites sont décrits. Puis, dans la section 4, nous décrivons le principe des chaînes d'automates et leur application au diagnostic. L'utilisation conjointe des deux techniques que nous proposons dans cet article est décrite dans la section 5, et une application présentée dans la section 6. Enfin, nous présentons le principe d'une généralisation de ce résultat dans la section 7.

2 Automates et diagnostic

Cette section présente rapidement les définitions d'automates et leur application au diagnostic. Nous reprenons ici la notation introduite en [5].

2.1 Automates et trajectoires

Nous représentons les comportements par des automates, définis de manière traditionnelle :

Définition 1 (Automate)
On appelle automate *le tuple* $A = (Q, E, T, I, F)$ *où :*
- Q *est un ensemble d'états,*
- E *est un ensemble d'événements,*
- $T \subseteq (Q \times 2^E \times Q)$ *est un ensemble de transitions* $t = (q, l, q')$ *où* q *est l'état de départ,* q' *l'état d'arrivée et* l *un ensemble d'événements* ($l \subseteq E$),
- I *est un ensemble d'états initiaux* ($I \subseteq Q$),
- F *est un ensemble d'états finaux* ($F \subseteq Q$).

Pour tout état $q \in Q$, on considère que la transition $t = (q, \{\}, q')$ est une transition de l'automate ($t \in T$). L'ensemble d'événements l associé à une transition $t = (q, l, q')$ est appelée *étiquette* sur l'ensemble d'événements E.

Définition 2 (Chemin)
On appelle chemin *sur l'automate* $A = (Q, E, T, I, F)$ *une double séquence d'états et de transitions* $q_0 \xrightarrow{l_1} \ldots \xrightarrow{l_n} q_n$ *telle que* $\forall i \in \{1, \ldots, n\},\ (q_{i-1}, l_i, q_i) \in T$.

Un chemin sur un automate est une succession de transitions. On appelle *trajectoire* un chemin partant d'un état initial $q_0 \in I$ et menant à un état final $q_n \in F$. Le but d'un automate est de représenter l'ensemble des trajectoires qui décrivent les comportements possibles du système (voir le paragraphe 2.2 d'application au diagnostic). Pour cette raison, on considère que deux automates A_1 et A_2 sont identiques s'ils ont le même ensemble de trajectoires. La *simplification* de l'automate A est l'opération consistant à supprimer les transitions et les états non accessibles depuis un état initial ou ne conduisant pas à un état final. Par la suite, les automates calculés sont implicitement simplifiés (calculer directement l'automate simplifié est généralement plus simple et produit un automate équivalent mais plus petit).

Définition 3 (Synchronisation d'étiquettes)
Soit l_1 *une étiquette sur* E_1 *et* l_2 *une étiquette sur* E_2. *On dit que* l_1 *et* l_2 *sont* synchrones *ssi* $l_1 \cap (E_1 \cap E_2) = l_2 \cap (E_1 \cap E_2)$. *Leur synchronisation notée* $\Theta(l_1, l_2)$ *est l'étiquette* $l_1 \cup l_2$ *sur l'ensemble d'événements* $E_1 \cup E_2$.

Deux étiquettes sont synchrones si les événements de synchronisation ($E_1 \cap E_2$) présents dans l'une des étiquettes sont également présents dans l'autre.

Définition 4 (Synchronisation d'automates)
Soient deux automates $A_1 = (Q_1, E_1, T_1, I_1, F_1)$ *et* $A_2 = (Q_2, E_2, T_2, I_2, F_2)$. *On appelle* automate synchronisé *de* A_1 *et* A_2, *noté* $A_1 \otimes A_2$, *l'automate* $A = (Q, E, T, I, F)$ *défini par :*
- $Q = Q_1 \times Q_2$,
- $E = E_1 \cup E_2$,
- $T = \{((q_1, q_2), l, (q_1', q_2')) \mid (q_1, l \cap E_1, q_1') \in T_1 \wedge (q_2, l \cap E_2, q_2') \in T_2\}$,
- $I = I_1 \times I_2$,
- $F = F_1 \times F_2$.

La synchronisation consiste à emprunter deux transitions dont les étiquettes sont synchrones simultanément sur les deux automates.

2.2 Application au diagnostic

Nous nous plaçons dans le cadre des *systèmes réactifs* [4, 6]. Le système réagit à la suite de l'occurrence d'événements. Certains événements peuvent être simultanés. C'est le cas si l'on considère que l'occurrence d'un événement peut déclencher de manière synchrone d'autres événements en cascade. Le comportement du système peut alors être décrit par un automate. Chacune des trajectoires sur l'automate représente l'une des évolutions possibles du système. Chaque transition de l'automate est étiquetée par les événements ayant provoqué l'évolution de l'état et les observations émises le cas échéant. Par ailleurs, il n'y a pas de contrainte sur l'état final du système. C'est pourquoi on considère que tous les états sont finaux.

Définition 5 (Modèle)
Le modèle *d'un système est un automate* $Mod = (Q, E, T, I, F)$ *où* $F = Q$.

Les observations émises par le système sont reçues par le superviseur (dispositif en charge du diagnostic).

Dans les systèmes réels, les observations reçues ne permettent généralement pas de connaître précisément les observations émises et l'ordre de leur émission. Ceci est dû aux canaux de communication entre les capteurs et le superviseur. En effet, ces canaux induisent un délai de transmission non négligeable et peuvent même parfois perdre des observations. De plus, les capteurs n'étant pas synchronisés, il n'est pas toujours possible de déterminer l'ordre d'émission des observations étant donnée leur date d'émission qui dépend de l'horloge du capteur. Aussi, étant donnée une séquence d'observations reçues, il existe un ensemble de séquences potentielles d'observations émises. Ces séquences peuvent être assimilées à des trajectoires et sont donc représentées par un automate.

Définition 6 (Automate d'observations)
*L'*automate des observations *est un automate* $Obs = (Q, E, T, I, F)$.

Cette représentation des observations est comparable à l'utilisation des *index space* dans les travaux de G. Lamperti et M. Zanella [6] ou la représentation sous forme de langage des observations dans les travaux de Console et coll. [7]. On peut également citer l'utilisation d'automate pour représenter les formules logiques dans le cadre de la vérification de modèles [8].
L'ensemble des événements observables est l'intersection entre les événements de l'automate des observations et les événements du modèle : $\mathcal{O} = E_{Obs} \cap E_{Mod}$.
Dans la suite de ce papier, nous considérons que l'automate des observations est construit automatiquement. Nous ne nous intéressons pas à la manière dont il est construit.

Définition 7 (Diagnostic)
Le diagnostic *d'un système, noté* Δ_n*, est un automate décrivant les trajectoires possibles sur le modèle du système compatibles avec les observations reçues par le superviseur durant la période* $[t_0, t_n]$.

Le diagnostic peut être calculé par synchronisation du modèle et de l'automate des observations :

$$\Delta = Mod \otimes Obs \tag{1}$$

Une difficulté qui se pose à partir de ce résultat est qu'il est parfois impossible de construire les automates *Mod* et Δ. En effet, ces automates représentent le comportement du système, et celui-ci peut avoir un certain nombre d'exécutions en parallèle (dites *concurrentielles* ou *indépendantes*). Dans ce cas, le nombre d'états de l'automate peut grossir de manière exponentielle avec le nombre de composants dont est formé le système. L'approche que nous présentons dans la section suivante vise à contenir cette *explosion du nombre d'états* par une modélisation et un diagnostic *décentralisés*.

3 Diagnostic décentralisé

Dans les systèmes réels, l'automate modélisant le système est trop grand pour être calculé. C'est pour cette raison qu'on considère généralement une modèlisation décentralisée. Les algorithmes décentralisés de diagnostic permettent également de réduire la taille de l'automate produit. On peut se référer à [4] pour plus de détails.
Dans un premier temps, nous présentons la modélisation décentralisée et le diagnostic décentralisé. Ensuite nous expliquons pourquoi il est possible de conserver des diagnostics de sous-systèmes indépendants et pourquoi cela est intéressant.

3.1 Modélisation décentralisée

La plupart des systèmes réels sont constitués de composants interconnectés communiquant entre eux. De même que chaque composant est fabriqué séparément, il est possible de modéliser le comportement de chacun de ces composants séparément.
Le comportement d'un composant $\mathcal{C}_i$ est un automate $Mod_i = (Q_i, E_i, T_i, I_i, F_i)$ où $\mathcal{O}_i \subset E_i$ est l'ensemble des événements observable qui conduisent à l'émission d'une observation par le composant. Chaque composant est un système réactif, c'est-à-dire qu'il réagit à l'occurrence d'événements extérieurs. Ainsi, chaque transition est étiquetée par un événement exogène ou un message provenant d'un autre composant, et éventuellement par des messages envoyés à d'autres composants et des observations. L'envoi de messages entre composants est instantané (communications synchrones). Dans le cas contraire, on considère les connexions comme des composants du système, et il est nécessaire de les modéliser.
Le modèle global Mod_{ss} d'un (sous-)système *ss* constitué des composants $\mathcal{C}_1, \ldots, \mathcal{C}_n$ peut être obtenu par synchronisation des modèles : $Mod_1 \otimes \ldots \otimes Mod_n$.
Remarquons cependant qu'on considère généralement qu'un seul événement exogène peut avoir lieu à un instant donné. Ainsi, il est nécessaire de supprimer les transitions de Mod_{ss} comprenant plusieurs événements exogènes. Nous laissons cependant ce point de côté pour cette présentation. Le nombre d'états du modèle global est en $o(x^n)$ où x est le nombre d'états par modèle de composant et n le nombre de composants. Aussi, on ne calcule généralement pas le modèle explicite du système. Le modèle décentralisé du système est l'ensemble des modèles des composants : $\{Mod_1, \ldots, Mod_n\}$.

3.2 Diagnostic décentralisé

De même que le modèle du système, il est souvent possible, grâce aux hypothèses sur le système, de représenter l'automate des observations comme la synchronisation d'automates d'observations Obs_i portant sur chacun des composants $\mathcal{C}_i$: $Obs = Obs_1 \otimes \ldots \otimes Obs_n$.

Alors, il est possible de réécrire le diagnostic de la manière suivante : $\Delta = Mod \otimes Obs = (Mod_1 \otimes \ldots \otimes Mod_n) \otimes (Obs_1 \otimes \ldots \otimes Obs_n) = (Mod_1 \otimes Obs_1) \otimes \ldots \otimes (Mod_n \otimes Obs_n)$. Ce résultat est vrai à un renommage des états près.

Le *diagnostic du composant* $\mathcal{C}_i$ est noté $\Delta_i = Mod_i \otimes Obs_i$. Ce diagnostic contient les trajectoires possibles sur le modèle du composant $\mathcal{C}_i$ étant données les observations sur ce composant. Ce diagnostic fait des hypothèses sur les messages échangés entre ce composant et les autres composants. Pour vérifier les hypothèses, il est nécessaire d'effectuer la *fusion* (c'est-à-dire la synchronisation) des diagnostics locaux. La fusion des diagnostics est généralement effectuée deux à deux jusqu'à obtenir le diagnostic *global* du système.

La synchronisation de deux automates produit le moins d'états lorsque les automates ont beaucoup d'événements de synchronisation. On voit qu'il est donc recommandé de fusionner les diagnostics selon une *stratégie* visant à produire le moins possible d'états dans les diagnostics intermédiaires (voir [4]).

Si le système produit beaucoup d'observations et qu'il est extrêmement synchronisé, il est raisonnable de penser que la fusion des diagnostics produira un diagnostic global Δ de taille relativement réduite. Quand cette hypothèse n'est pas respectée, il est cependant possible de ne pas fusionner les diagnostics de sous-systèmes comme nous le présentons dans le paragraphe suivant.

3.3 Intérêts du diagnostic décentralisé

Définition 8 (Entrelacement)

Soit $p' = q'_0 \xrightarrow{l'_1} \ldots \xrightarrow{l'_{n1}} q'_{n1}$ *et* $p'' = q''_0 \xrightarrow{l''_1} \ldots \xrightarrow{l''_{n2}} q''_{n2}$ *deux chemins. Un chemin* p *est un* entrelacement *de* p' *et* p'' *si on a récursivement :*

- $n1 = 0 \Rightarrow p = (q'_0, q''_0) \xrightarrow{l''_1} \ldots \xrightarrow{l''_{n2}} (q'_0, q''_{n2})$,
- $n2 = 0 \Rightarrow p = (q'_0, q''_0) \xrightarrow{l'_1} \ldots \xrightarrow{l'_{n1}} (q'_{n1}, q''_0)$,
- *sinon*
  - $p = (q'_0, q''_0) \xrightarrow{l'_1} p^1$ *où* p^1 *est un entrelacement de* $q'_1 \xrightarrow{l'_2} \ldots \xrightarrow{l'_{n1}} q'_{n1}$ *et* p'', *ou*
  - $p = (q'_0, q''_0) \xrightarrow{l''_1} p^2$ *où* p^2 *est un entrelacement de* p' *et de* $q''_1 \xrightarrow{l''_2} \ldots \xrightarrow{l''_{n2}} q''_{n2}$.

On voit que l'entrelacement de deux chemins consiste à considérer chacun des comportements de deux chemins de manière concurrentielle. L'entrelacement de deux chemins est très facile à calculer, mais le nombre de possibilités est $\frac{(n1+n2)!}{n1!\ n2!}$. On considère donc les deux trajectoires p_1 et p_2 séparément plutôt que l'ensemble des trajectoires p.

Ainsi, il existe six entrelacements de $q'_0 \xrightarrow{a_1} q'_1 \xrightarrow{a_2} q'_2 \xrightarrow{a_3} q'_3$ et $q''_0 \xrightarrow{b_1} q''_1 \xrightarrow{b_2} q''_2$ dont l'un d'eux est $(q'_0, q''_0) \xrightarrow{b_1} (q'_0, q''_1) \xrightarrow{b_2} (q'_0, q''_2) \xrightarrow{a_1} (q'_1, q''_2) \xrightarrow{a_2} (q'_2, q''_2) \xrightarrow{a_3} (q'_3, q''_2)$.

Considérons un système de deux composants avec : $\Delta_1 = Mod_1 \otimes Obs_1$ et $\Delta_2 = Mod_2 \otimes Obs_2$, tels qu'il n'y a aucune occurrence d'un événement de synchronisation sur chacun des automates Δ_1 et Δ_2 (propriété d'*indépendance*) et $\Delta = \Delta_1 \otimes \Delta_2$. Alors, on a le résultat suivant :

Résultat 1

Pour toute trajectoire traj de Δ, *il existe une trajectoire* $traj_1$ *de* Δ_1 *et une trajectoire* $traj_2$ *de* Δ_2 *telles que la trajectoire traj est un entrelacement de* $traj_1$ *et* $traj_2$. *Réciproquement, pour toute paire de trajectoires* $traj_1$ *de* Δ_1 *et* $traj_2$ *de* Δ_2, *tout entrelacement traj de* $traj_1$ *et de* $traj_2$ *est une trajectoire de* Δ.

Ce résultat découle simplement de la définition de synchronisation d'automates.

L'ensemble des trajectoires de Δ est l'ensemble des trajectoires obtenues par entrelacement de deux trajectoires de Δ_1 et Δ_2. Il est donc inutile, dans le cadre du diagnostic, de synchroniser Δ_1 et Δ_2.

Dans le résultat présenté ci-dessus, Δ_i correspond au diagnostic d'un sous-système noté Γ_i. On peut généraliser ce résultat à un nombre quelconque de sous-systèmes.

Ce résultat est intéressant mais nécessite une hypothèse forte : les sous-systèmes ne doivent pas avoir communiqué durant la période que l'on cherche à diagnostiquer. Ceci n'est généralement pas le cas. C'est pourquoi nous voulons profiter des chaînes d'automates et du découpage temporel pour mettre en évidence des périodes d'indépendance entre les sous-systèmes et réduire la taille de l'automate représentant le diagnostic.

4 Diagnostic par morceaux

Cette section présente rapidement l'utilisation des chaînes d'automates dans le cadre du diagnostic. On pourra se référer à [5, 9] pour plus d'informations sur ce sujet.

Une chaîne d'automates $\mathcal{E}_A$ est une séquence d'automates $(A^1, \ldots, A^n)$. Une trajectoire sur $\mathcal{E}_A$ est une séquence de trajectoires $traj^i$ sur chacun des automates A^i telles que le dernier état de la trajectoire $traj^i$ est le premier état de la trajectoire $traj^{i+1}$. Soit $\forall i,\ traj^i = q^i_0 \xrightarrow{l^i_1} \ldots \xrightarrow{l^i_{mi}} q^i_{mi}$ (avec $q^i_{mi} = q^{i+1}_0$), alors $q^1_0 \xrightarrow{l^1_1} \ldots \xrightarrow{l^1_{m1}} q^1_{m1} \xrightarrow{l^2_1} q^2_1 \xrightarrow{l^2_2} \ldots \xrightarrow{l^n_{mn}} q^n_{mn}$ est une trajectoire de la chaîne.

On dit qu'une chaîne $\mathcal{E}_A$ est un *découpage correct* de l'automate A, noté $A = Sli^{-1}(\mathcal{E}_A)$ si toute trajectoire de $\mathcal{E}_A$ est une trajectoire de A et réciproquement. La figure 1 présente un exemple d'automate et un découpage correct de celui-ci (les étiquettes de transitions ne sont pas représentées pour simplifier la figure). Les états initiaux sont représentés par les transitions entrantes et les états finaux par des doubles cercles. On

voit par exemple que la trajectoire $1 \longrightarrow 3 \longrightarrow 5 \longrightarrow 4 \longrightarrow 6$ est présente sur l'automate et la chaîne d'automates.

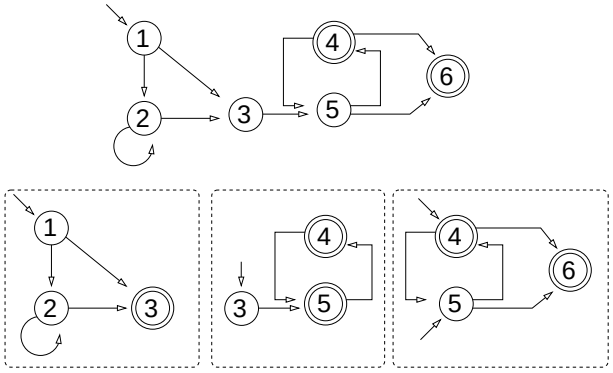

Figure 1: Automate et découpage.

Dans [5], nous considérons que l'automate d'observations *Obs* a été découpé en une chaîne $(Obs^1, \ldots, Obs^n)$. Ce découpage a été effectué par fenêtres temporelles, c'est-à-dire que si l'on considère que *Obs* représente les observations émises pendant la période $[t_0, t_n]$, alors Obs^i représente les observations émises pendant $[t_{i-1}, t_i]$.
On définit $Mod \otimes \mathcal{E}_{Obs} = (Mod \otimes Obs^1, Mod^- \otimes Obs^2, \ldots, Mod^- \otimes Obs^n)$ où Mod^- représente la fermeture suffixe de *Mod* (tous les états de Mod^- sont initiaux). Nous avons montré dans [5] que $\mathcal{E}_\Delta = Mod \otimes \mathcal{E}_{Obs}$ est un découpage correct de $\Delta = Mod \otimes Obs$. Ce calcul est appelé *diagnostic par morceaux*, le *morceau* $\Delta^i = Mod^- \otimes Obs^i$ (sauf $\Delta^1 = Mod \otimes Obs^1$) étant le diagnostic de la période $[t_{i-1}, t_i]$. Nous avons également montré qu'il est nécessaire, si le modèle est grand, de *restreindre* l'ensemble des états initiaux de Δ^i par l'ensemble des états finaux de Δ^{i-1}, c'est-à-dire de ne considérer comme initiaux que les états finaux à la précédente fenêtre. Cette opération est appelée *I-raffinement*, mais nous l'appelons simplement *raffinement* par la suite. On définit donc : $\Delta^i = (Mod^- \otimes Obs^i)[F_\Delta^{i-1}]$ où F_Δ^{i-1} est l'ensemble des états finaux de Δ^{i-1}. Cette restriction permet de supprimer un grand nombre de trajectoires sur les morceaux d'automates et, par simplification de ces morceaux, un grand nombre d'états et de transitions.
À partir d'un découpage correct d'un automate, il est immédiat de retrouver (par *reconstruction*) l'automate d'origine. Nous avons montré en [5] qu'il n'était pas nécessaire de reconstruire le diagnostic une fois obtenu le diagnostic par morceaux.

5 Diagnostic par morceaux et diagnostic décentralisé

Dans cette section, nous proposons notre approche qui consiste à utiliser le diagnostic décentralisé sur les morceaux de diagnostic obtenus grâce à la méthode par chaînes d'automates. On peut noter que les exposants sont réservés aux *morceaux* des chaînes d'automates, tandis que les indices concernent les automates associés à des sous-systèmes.
Tout d'abord, sont rappelées les limites des deux méthodes séparément pour limiter l'explosion du nombre d'états et est présenté le principe de notre méthode. Le raffinement est ensuite utilisé pour réduire le nombre d'états du diagnostic de chaque fenêtre, et finalement l'ensemble des états finaux est calculé pour permettre le raffinement de la fenêtre suivante.

5.1 Limites des deux méthodes présentées

Il apparait qu'utiliser le diagnostic par sous-systèmes permet de réduire considérablement la complexité du calcul. Quand Δ_1 et Δ_2 sont indépendants, le diagnostic $\Delta = \Delta_1 \otimes \Delta_2$ (où Δ_i est le diagnostic du sous-système ss_i) comporte en effet $n_1 \times n_2$ états où n_i est le nombre d'états de Δ_i. Cependant, lorsque le diagnostic est effectué sur une longue période, il est peu probable que la propriété d'indépendance soit respectée. En effet, tôt ou tard, les sous-systèmes communiquent entre eux, et leurs diagnostics doivent alors être synchronisés (et donc, les *parties* indépendantes aussi).
Considérons ainsi l'exemple de la figure 2. Les transitions de ces automates ne comprennent ici qu'un seul événement. Dans cet exemple, le seul événement de synchronisation est c. Lors de la synchronisation (voir figure 3), la trajectoire étiquetée par b_6 et b_7 du second automate disparaît puisqu'elle ne peut pas se synchroniser avec l'événement c du premier automate. En effet, le premier automate indique que l'événement c a forcément eu lieu (puisque la seule trajectoire passe par la transition étiquetée par c). Donc, les trajectoires possibles sur le second automate doit comporter une transition étiquetée par c. La synchronisation apporte donc une information indispensable au diagnostic. On voit nettement que les comportements des deux sous-systèmes sont souvent concurrentiels, et cela induit un nombre important d'états dans le diagnostic global.

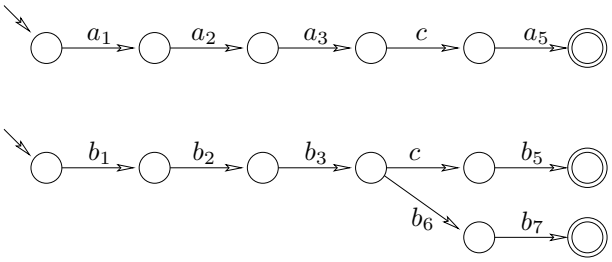

Figure 2: Diagnostics de deux sous-systèmes.

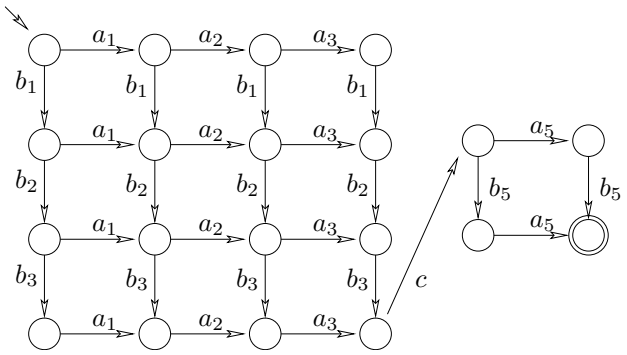

Figure 3: Diagnostic global de la figure 2 (événement c synchronisé).

D'un autre côté, les chaînes de diagnostic ne permettent également pas, seules, de réduire la complexité du calcul. En effet, même en utilisant les opérations de raffinement, le nombre total d'états de la chaîne de diagnostics est supérieur au nombre d'états du diagnostic global (les états *frontière* étant en double, voir la figure 1).

Il est cependant possible d'utiliser les deux méthodes conjointement. En effet, nous avons vu qu'il n'est pas possible d'utiliser l'indépendance pour des diagnostics sur des durées trop longues. En revanche, les chaînes permettent de construire le diagnostic sur des périodes successives *courtes*. On peut alors espérer tirer parti de l'indépendance des sous-systèmes durant certaines périodes.

5.2 Notre approche

Nous reprenons l'exemple précédent. Nous proposons une représentation comme dans la figure 4. Puisque durant la première fenêtre temporelle, les deux sous-systèmes ne communiquent pas, le diagnostic global du système pour cette période n'est pas calculé, mais il est remplacé par les diagnostics des deux sous-systèmes. Durant la deuxième fenêtre, les sous-systèmes ont effectivement communiqué. Nous les synchronisons donc. Enfin, dans la dernière fenêtre, il est à nouveau possible de considérer les deux sous-systèmes séparement.

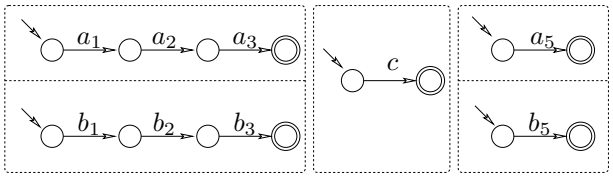

Figure 4: Diagnostic utilisant chaînes et concurrence.

5.3 Raffinement

Le raffinement est une étape indispensable du diagnostic. Elle consiste à restreindre l'ensemble des états initiaux d'un morceau d'une chaîne par l'ensemble des états finaux du morceau précédent. Cette opération ne modifie pas l'ensemble de trajectoires de la chaîne d'automates mais réduit la taille des automates. Nous faisons les remarques suivantes :

1. Tout d'abord, il n'est réellement possible de garder le diagnostic sous forme de chaînes d'automates que si celle-ci est raffinée. En effet, dans le cas contraire, il y a beaucoup d'états et de transitions inutiles. Il est alors impossible de raisonner sur ces automates puisque la plupart des transitions et des états pourraient être supprimées. L'opération de raffinement doit donc être conservée par la méthode que nous proposons pour obtenir l'ensemble des états à la date t_i.
2. D'autre part, le raffinement permet de réduire de manière très importante le nombre d'états des morceaux de diagnostic quand le modèle du système est grand. Or, nous nous plaçons dans un contexte où il est indispensable de considérer le modèle de manière décentralisée. Le nombre d'états du modèle est donc très important. Il faut donc appliquer le raffinement pour réduire le nombre d'état des morceaux de diagnostic.
3. Enfin, ne pas raffiner les morceaux de diagnostic conduit à un nombre plus important de transitions dans le diagnostic. Ces transitions supplémentaires peuvent comporter des événements de synchronisation et les comportements décrits par les automates peuvent ne plus être indépendants. Ainsi, la probabilité d'être assuré de la propriété d'indépendance entre deux sous-systèmes est réduite, et on perd beaucoup de l'intérêt de cette méthode.

Les trois points ci-dessus montrent qu'il est indispensable de raffiner la chaîne de diagnostic, et donc de connaître et d'utiliser l'ensemble des états finaux du diagnostic Δ^i, ou au moins un sur-ensemble le plus petit possible.

Soit I l'ensemble des états à t_i (I est l'ensemble des états finaux de Δ^i). Alors, le diagnostic de la période $[t_i, t_{i+1}]$ est le suivant : $\Delta^{i+1} = (Mod^- \otimes Obs^{i+1})[I]$.
On considère qu'il y a m composants et que les observations Obs^{i+1} peuvent s'écrire de la manière suivante : $Obs^{i+1} = Obs_1^{i+1} \otimes \ldots \otimes Obs_m^{i+1}$ où chaque automate Obs_j^{i+1} contient des observations provenant du composant j.
Alors, le diagnostic peut être écrit ainsi : $\Delta^{i+1} = (\Delta_1^{i+1} \otimes \ldots \otimes \Delta_m^{i+1})[I]$ avec $\Delta_j^{i+1} = Mod_j^- \otimes Obs_j^{i+1}$.
Δ_j^{i+1} est le diagnostic local du composant j pendant la période $[t_i, t_{i+1}]$. Nous remarquons que nous considérons ici que tous les états de Mod_j sont potentiellement

initiaux. Or, il est possible de restreindre cet ensemble à l'ensemble des états du composant j dans I.

Définition 9 (Projection)
Soit Q, un ensemble d'états q d'un système ($q = (q_1, \ldots, q_m)$ tel que q_i est un état de A_i). On note $Q \downarrow \{A_{j1}, \ldots, A_{jp}\}$, appelé projection *de l'ensemble d'états Q sur $\{A_{j1}, \ldots, A_{jp}\}$, l'ensemble des états $(q_{j1}, \ldots, q_{jp})$ avec q_{jk} un état de A_{jk} tel qu'il existe $q \in Q$ avec $q = (q_1, \ldots, q_{j1}, \ldots, q_{jp}, \ldots, q_m)$.*

La projection consiste à ne garder que les états des automates qui nous intéressent (projection de l'état du système sur un sous-système par exemple). Alors, on peut facilement prouver :

Résultat 2
Soit $\Delta_j^{i+1} = (Mod_j^- \otimes Obs_j^{i+1})[I \downarrow \{Mod_j, Obs_j^{i+1}\}]$. Alors $\Delta^{i+1} = (\Delta_1^{i+1} \otimes \ldots \otimes \Delta_m^{i+1})[I]$

La preuve de ce résultat est donnée en annexe.
Considérons à présent que nous voulons fusionner les diagnostics des composants 1 à k. Alors, le diagnostic de ce sous-système ss est obtenu par $\Delta_{ss}^{i+1} = \Delta_1^{i+1} \otimes \ldots \otimes \Delta_k^{i+1}$. Cependant, pour les mêmes raisons que précédemment, il est possible de restreindre les états initiaux de ce diagnostic de la manière suivante : $\Delta_{ss}^{i+1} = (\Delta_1^{i+1} \otimes \ldots \otimes \Delta_k^{i+1})[I \downarrow \{Mod_1, \ldots, Mod_k, Obs_1^{i+1}, \ldots, Obs_k^{i+1}\}]$.

5.4 Calcul des états finaux

Nous avons calculé le diagnostic pour chaque composant et montré comment il est possible de raffiner le diagnostic de sous-systèmes pour une fenêtre étant donné l'ensemble des états finaux de la précédente fenêtre. Nous montrons à présent comment calculer l'ensemble des états finaux de cette fenêtre pour permettre le raffinement de la fenêtre suivante. Nous considérons que nous avons fusionné les diagnostics pour k sous-systèmes indépendants. Les composants de chaque sous-système j sont notés $j1, \ldots, jn$. La propriété indiquant qu'il existe une trajectoire de l'état q à l'état q' est notée $q \rightsquigarrow q'$. L'ensemble F est alors calculé par :

Résultat 3
L'ensemble des états finaux de la fenêtre temporelle $[t_i, t_{i+1}]$ est : $F = \{q' \mid \exists q \in I,\ \forall j \in \{1, \ldots, k\},\ (q \downarrow \{Mod_{j1}, \ldots, Mod_{jn}, Obs_{j1}^{i+1}, \ldots, Obs_{jn}^{i+1}\}) \rightsquigarrow (q' \downarrow \{Mod_{j1}, \ldots, Mod_{jn}, Obs_{j1}^{i+1}, \ldots, Obs_{jn}^{i+1}\})\}$.

Ce résultat est prouvé en annexe. Soit q un état initial de I et q_j la projection de q sur $\{Mod_{j1}, \ldots, Mod_{jn}, Obs_{j1}^{i+1}, \ldots, Obs_{jn}^{i+1}\}$ (c'est-à-dire l'état du sous-système j). Alors, on prend pour chaque j un état q'_j tel qu'il existe une trajectoire de q_j à q'_j. L'état $q' = (q'_1, \ldots, q'_k)$ est un état de F. L'ensemble des états finaux est l'ensemble des états q' qui peuvent être obtenus de cette manière.

Remarquons que la taille de l'ensemble des états finaux est exponentielle par rapport au nombre de composants du système. Ce calcul de l'ensemble des états finaux peut être très coûteux et l'énumération des états peut être de complexité équivalente au calcul du diagnostic global. Cependant, il semble difficile d'éviter ce calcul et de considérer une représentation implicite. En effet, imaginons un système constitué de composants dont on est sûr à l'issue de la première fenêtre temporelle qu'un des composants est en panne (sans savoir lequel, voir figure 5). Sur la figure, O représente l'état *ok* et P l'état *en panne*, l'état du second composant est primé. Considérons que dans une fenêtre temporelle suivante, les deux composants évoluent indépendamment sans changer d'états. Dans le diagnostic local, l'état des composants est *ok* ou *en panne*. Au début de la troisième fenêtre, et sans énumération explicite des états finaux, il est nécessaire de retourner à la première fenêtre pour savoir que le système n'a pas ses deux composants à l'état *ok*. Le calcul devient particulièrement ardu lorsque les sous-systèmes indépendants varient d'une fenêtre temporelle à une autre.

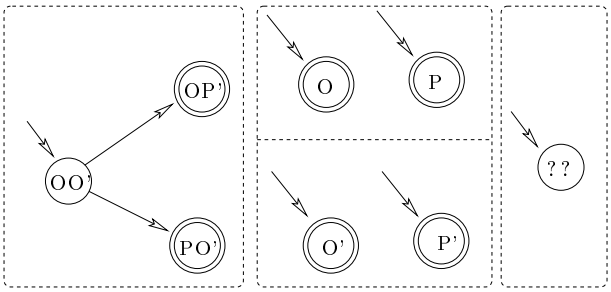

Figure 5: Quels sont les états initiaux de la troisième fenêtre ?

6 Résultats et expérimentations

Dans cette section, nous présentons une expérimentation de l'utilisation conjointe du diagnostic décentralisé et du diagnostic par morceaux.

6.1 Algorithme

Nous avons effectué l'expérimentation en utilisant l'algorithme 1. L'implémentation a été faite en Java et l'algorithme utilisé sur un processeur Intel Pentium-4 2,40GHz, 1 GB RAM, sous Linux.
L'algorithme prend en entrée un ensemble d'états initiaux I^1 qui correspondent aux états initiaux du modèle. À chaque fenêtre, on calcule d'abord les diagnostics locaux. Ensuite, les diagnostics sont fusionnés tant qu'il est nécessaire de les fusionner. Notons que lorsque le diagnostic $\Delta_{k,l}^i$ est calculé, les diagnostics Δ_k^i et Δ_l^i sont supprimés de la liste des diagnostics.

Algorithme 1 L'algorithme de diagnostic

Entrée : I^1 états initiaux
pour chaque fenêtre i **faire**
  /* *Diagnostics locaux* */
  pour chaque composant c_j **faire**
    $\Delta^i_j = (Mod^-_j \otimes Obs^i_j)[I^i \downarrow \{Mod_j, Obs^i_j\}]$
  fin pour
  /* *Fusion des diagnostics* */
  tant que il est nécessaire de synchroniser deux sous-systèmes ss_k et ss_l **faire**
    $\Delta^i_{k,l} = (\Delta^i_k \otimes \Delta^i_l)[I^i \downarrow \{ss_k, ss_l\}]$
  fin tant que
  /* *Calcul de l'ensemble des états finaux* */
  $I^{i+1} = final(I^i, \{\Delta_{ss_1}, \ldots, \Delta_{ss_j}\})$
fin pour

Enfin, l'ensemble des états finaux est calculé comme présenté dans le résultat 3.

6.2 Système étudié

Le système considéré est un réseau de composants interconnectés comme présenté dans la figure 6. Deux composants sont dits voisins s'il existe une connexion entre ceux-ci.

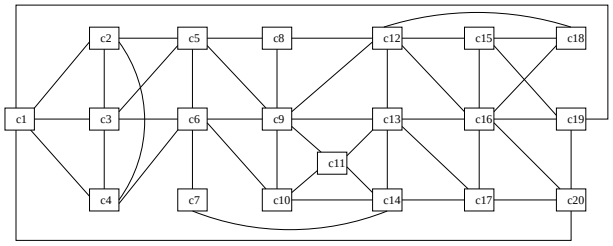

Figure 6: Topologie du système.

Chacun des composants a le même fonctionnement. Lorsqu'une panne a lieu sur un composant, celui-ci demande à tous ses voisins de redémarrer et essaie de redémarrer. Lorsqu'un composant vient de subir une panne et reçoit un message de redémarrage, il ignore le message. En revanche, s'il est en train de redémarrer, il redémarre à nouveau. Le début et la fin de redémarrage sont observables. La figure 7 présente le modèle de manière simplifiée d'un composant.
Les états O, F et R correspondent respectivement à OK, en panne (*Faulty*) et en redémarrage (*Reboot*). Sur l'étiquette de transition, l'événement avant la flèche est l'événement déclencheur et les événements après sont les observations et les messages envoyés aux autres composants. L'événement *reboot ?* (resp. *reboot !*) indique la réception (resp. l'émission) du message *reboot* depuis (resp. vers) un composant connexe. L'unique événement de faute est *fault*. Les observables sont *IReboot* et *IAmBack*. Les événements *eor* et *eof*

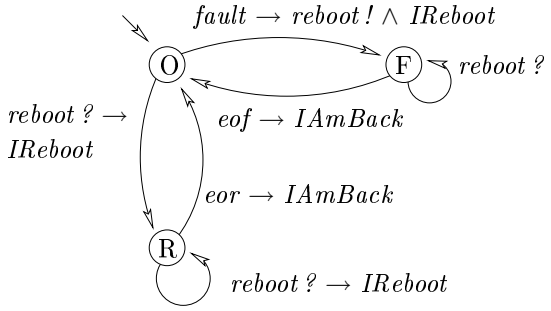

Figure 7: Modèle d'un composant.

correspondent à *end of reboot* et *end of fault*.
La difficulté du diagnostic réside dans le fait que lorsqu'un composant tombe en panne, ou lorsqu'un de ses voisins tombe en panne, le composant émet les mêmes observations (*IReboot* suivi de *IAmBack*). Aussi, il est nécessaire de raisonner de manière globale pour déterminer quel composant a subi la panne.
Si on cherche à construire le modèle du système, on obtient $|\{\mathrm{O},\mathrm{F},\mathrm{R}\}^{20} - 1| \approx 3,4.10^9$ états. Le modèle décentralisé ne comporte quant à lui que $3 \times 20 = 60$ états. Il est donc nécessaire d'utiliser le modèle décentralisé.

6.3 Mode opératoire

Les observations locales à chaque composant sont ordonnées et on y insère les *tics* d'horloge Υ_i (voir [5]) correspondant au passage d'une fenêtre temporelle à la fenêtre suivante. L'automate des observations locales est présenté figure 8.

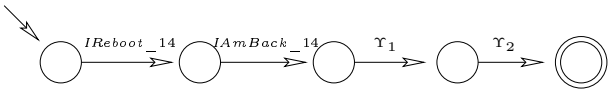

Figure 8: Automate des observations locales pour le composant numéro 14.

Les tics d'horloge se passent simultanément sur tous les composants. Il convient donc de les synchroniser lors de la fusion des diagnostics. Cependant, s'il n'y a pas d'autres événements de synchronisation, leur synchronisation lors de la fusion des diagnostics apporte comme seule information l'ordre des pannes (et ne discrimine pas la localisation des pannes). Aussi, pour déterminer si deux sous-systèmes ont un comportement indépendant, on ne considère pas les événements Υ_i.
Les morceaux d'automates de la chaîne d'observations pour chaque composant sont découpés grâce à Υ_i. Ainsi, l'automate Obs^1_{14} comporte trois états, reliés par des transitions étiquetées par *IReboot*_14 et *IAmBack*_14.

6.4 Expérimentation

La période à diagnotiquer comprend deux fenêtres temporelles. Durant la première fenêtre, les composants 1 et 11 tombent en panne forçant leurs voisins à redémarrer (19, 20, 2, 3 4 d'une part, 9, 10, 13, 14 d'autre part) puis reviennent en mode de fonctionnement normal, ainsi que leurs voisins. Durant la seconde période, le composant 6 tombe en panne et lui et ses voisins (3, 4, 5, 7, 9 et 10) redémarrent avec succès.
Avec la méthode de diagnostic classique, on voit qu'on ne peut pas diviser le sous-système constitué des 14 composants cités ci-dessus en sous-systèmes indépendants. Leurs diagnostics locaux doivent donc être fusionnés. On obtient ainsi 7 diagnostics de sous-systèmes, dont 6 diagnostics locaux à un composant. Le dernier diagnostic est un automate comprenant 2 275 états et 179 012 transitions.
La méthode utilisant les chaînes d'automates avec le diagnostic décentralisé produit 25 automates. Ces automates totalisent 274 états et 2 963 transitions. La taille nécessaire a donc été divisée par presque 100.

7 Généralisation du résultat

Dans cette section, nous présentons une généralisation possible du travail présenté dans cet article, mais sans la présenter formellement.
Une perspective possible de ce travail serait de généraliser l'imbrication entre le découpage et le calcul décentralisé. Par exemple, considérant un sous-système ss pendant une fenêtre temporelle $[t_i, t_{i+1}]$, est-il intéressant de découper la fenêtre pour ce sous-système (et pas les autres sous-systèmes) pour pouvoir considérer dans certaines de ces sous-fenêtres temporelles que les sous-systèmes $sss_1, \ldots, sss_k$ de ss sont indépendants ?
Ainsi, prenons la figure 4. Cette figure peut être abstraite comme présenté dans la partie supérieure de la figure 9. On voit qu'il y a trois fenêtres temporelles et que les première et dernière comportent deux sous-systèmes indépendants. On a vu que découvrir des comportements concurrentiels (traits horizontaux) permet de réduire fortement la taille du diagnostic. En bas de cette même figure, le diagnostic correspondant au sous-système supérieur durant la première fenêtre temporelle est découpé à une nouvelle date t permettant de considérer deux nouveaux sous-systèmes sss_1 et sss_2 pendant une des sous-fenêtres.
L'intérêt de cette méthode est donc de tirer à nouveau parti de l'aspect indépendant de sous-systèmes pendant certaines périodes. Cependant, la difficulté est de construire un algorithme capable de découvrir ces propriétés.

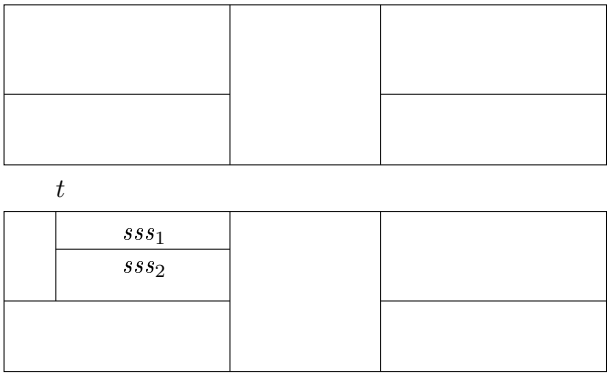

Figure 9: Imbrication de découpage et d'indépendance.

8 Conclusion et perspectives

Dans ce papier, nous avons montré qu'il était possible de tirer parti d'une représentation décentralisée du diagnostic comme présentée dans [4], ce qui est irréaliste dans le cas d'un diagnostic sur une période trop longue, en utilisant les chaînes d'automates présentées dans [5]. Nous avons montré qu'un point important était de calculer les ensembles d'états finaux de chaque fenêtre pour restreindre les états initiaux de la fenêtre suivante. Nous avons également montré de quelle manière il fallait les utiliser.
On peut citer deux autres approches de modélisation. La modélisation proposée dans [6] n'est pas décentralisée mais hiérarchique. Le problème qui apparaît est que la fusion des diagnostics ne peut s'effectuer que conformément à cette hiérarchie, et il n'est pas possible d'appliquer une stratégie de fusion. De plus, les sous-systèmes qui peuvent être considérés sont forcément statiques, alors que dans le cas général, il arrive souvent que les sous-systèmes indépendants changent d'une fenêtre temporelle à l'autre. L'approche de [10] présente une modélisation décentralisée *reconfigurable*, c'est-à-dire qu'il est possible de modifier les connexions entre les composants et d'ajouter ou supprimer des composants. Pour cela, la modélisation s'appuie sur une topologie explicite. Il est possible d'adapter le travail présenté dans le présent article pour [10] à la notation près.
Dans [4], il a été proposé l'utilisation des techniques d'ordre partiel (voir [11, 12]) pour empêcher l'explosion du nombre d'états lors de la fusion des diagnostics. Cette technique définit des traces, c'est-à-dire des ensembles de trajectoires équivalentes à un réordonnencement des événements indépendants près. Quand cela est possible, une seule trajectoire est conservée pour chaque trace. De cette manière, le nombre d'états est maîtrisé. Cependant, tous les états sont construits et, ensuite seulement, otés par simplification de l'automate représentant le diagnostic.

Comme nous l'avons vu à l'issue de la section 5, le calcul des états finaux d'une fenêtre temporelle peut être très coûteux. Est-il possible d'éviter cette énumération de manière efficace (notamment par BDD [13]) ?

Un autre point intéressant est l'utilisation du modèle présenté dans [10] pour permettre le calcul du diagnostic pendant les reconfigurations. Enfin, considérant que le découpage *en-ligne* des automates d'observations est difficile (il faut découper un automate qui n'est pas encore construit), il peut être intéressant de considérer un découpage de l'automate à une date t uniquement pour un sous-système et à une autre date t' pour les autres automates comme suggéré par la section 7.

A Preuve du résultat 2

Voici la preuve du résultat 2. Nous prouvons la propriété qui suit, et il est alors facile de généraliser ce théorème au résultat 2.
Soit $A = (A_1 \otimes A_2)[I]$. Soit $A' = (A_1[I \downarrow A_1] \otimes A_2[I \downarrow A_2])[I]$. Alors, $A = A'$.

Considérons une trajectoire $traj = q_0 \xrightarrow{l_1} \ldots \xrightarrow{l_k} q_k$ de A telle que $\forall i \in \{1, \ldots, k\}$, $q_i = (q_{i1}, q_{i2})$. Alors, puisque A est restreint sur I, $q_0 \in I$.
On a donc deux trajectoires $traj_j = q_{0j} \xrightarrow{l_{1j}} \ldots \xrightarrow{l_{kj}} q_{kj}$ (pour $j = \{1, 2\}$) telles que $traj_j$ est une trajectoire de A_j. Or, $q_{0j} \in (I \downarrow A_j)$ puisque $(q_{01}, q_{02}) \in I$. Donc, $traj_j$ est une trajectoire de $A_j[I \downarrow A_j]$.
Ainsi, $traj$ est une trajectoire de $(A_1[I \downarrow A_1] \otimes A_2[I \downarrow A_2])$. Et donc, $traj$ est une trajectoire de A'.

De même, toute trajectoire de A' est une trajectoire de A puisque la construction de A' comporte plus de restrictions que celle de A. Donc, $A = A'$.

Le résultat peut être étendu à m automates A_j. Si on note $A_j = Mod_j^- \otimes Obs_j^{i+1}$, on obtient le résultat 2.

B Preuve du résultat 3

Voici la preuve du résultat 3. Soit $\Delta^{i+1} = (\Delta_1^{i+1} \otimes \ldots \otimes \Delta_j^{i+1})[I]$ le diagnostic global du système. Alors, F est l'ensemble des états finaux de Δ^{i+1}.
On note $q = (q_1, \ldots, q_j)$ et $q' = (q'_1, \ldots, q'_j)$ deux états (q_k et q'_k sont les projections sur le sous-système k). Alors, $q' \in F \Leftrightarrow \exists q \in I$ tel qu'il existe une trajectoire sur Δ^{i+1} entre q et q'. Or, d'après le résultat 1, il vient qu'il existe une trajectoire entre q et q' si et seulement si $\forall k$ il existe une trajectoire entre q_k et q'_k sur Δ_k^{i+1}. Cette propriété est celle qui est indiquée dans le résultat 3.

Références

[1] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[2] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, pages 1555–1575, 1995.

[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis. Failure diagnosis using discrete-event models. In *IEEE Transactions on Control Systems Technology (CST-96)*, pages 105–124, 1996.

[4] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence Journal*, 164(1-2) :121–170, 2005.

[5] A. Grastien, M.-O. Cordier, and Ch. Largouët. First steps towards incremental diagnosis of discrete-event systems. In *Eighteenth Canadian Conference on Artificial Intelligence (AI'05)*, pages 170–181, Victoria (British-Columbia, Canada), May 2005. Springer-Verlag GmbH.

[6] G. Lamperti and M. Zanella. *Diagnosis of Active Systems*. Kluwer Academic Publishers, 2003.

[7] L. Console, Cl. Picardi, and M. Ribaudo. Diagnosis and diagnosability analysis using PEPA. In *Fourteenth European Conference on Artificial Intelligence (ECAI-00)*, pages 131–135, Berlin, Allemagne, 2000.

[8] M. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, pages 238–266, 1995.

[9] A. Grastien, M.-O. Cordier, and Ch. Largouët. Incremental diagnosis of discrete-event systems. In *Sixteenth International Workshop on Principles of Diagnosis (DX'05)*, pages 119–124, Pacific Grove, California, USA, June 2005.

[10] A. Grastien, M.-O. Cordier, and Ch. Largouët. Extending decentralized discrete-event modelling to diagnose reconfigurable systems. In *Fifteenth International Workshop on Principles of Diagnosis (DX'04)*, pages 75–80, Carcassonne (France), Jun 2004.

[11] Antoni W. Mazurkiewicz. Basic notions of trace theory. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop*, pages 285–363, London, UK, 1989. Springer-Verlag.

[12] D. Peled. On model checking using representatives. In *Fifth International Conference on Computer-Aided Verification (CAV-93)*, pages 409–423, Elounda, Grèce, Juin 1993.

[13] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8) :677–691, 1986.